ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Maestría En Seguridad Informática Aplicada

“IMPLEMENTACIÓN DE SEGURIDADES INFORMÁTICAS PARA UN SISTEMA WEB ORGANIZACIONAL DE ARCHIVOS**”**

EXAMEN DE GRADO (COMPLEXIVO)

Previa a la obtención del grado de:

MAGISTER EN SEGURIDAD INFORMÁTICA APLICADA

OSWALDO ADOLFO GARCÍA CLAVIJO

GUAYAQUIL – ECUADOR

AÑO: 2016

AGRADECIMIENTO

A Dios, por la oportunidad de cumplir esta meta en mi vida. A mis padres y hermanos, por el apoyo y las fuerzas para seguir adelante. A mis amigos, que me enseñaron a no rendirme.

OSWALDO ADOLFO GARCIA CLAVIJO

0917393449

DEDICATORIA

El presente trabajo va dedicado a mis amigos que supieron en todo momento darme ánimos para perseverar y no rendirme: Marcelo T., Martín M., Lorena M., Gabriel C., Víctor L., Ricardo N., Sonia M., Juan Carlos C., John P., Michelle P., Andrea Q., Manuel B., Marcia A., Bolívar V. Y en especial a mi madre, Luisa Clavijo.

OSWALDO ADOLFO GARCIA CLAVIJO

0917393449

TRIBUNAL DE SUSTENTACIÓN

_____

Ing. Lenin Freire

PROFESOR DELEGADO POR LA UNIDAD ACADEMICA

_____

Ing. Karina Astudillo

PROFESOR DELEGADO POR LA UNIDAD ACADEMICA

RESUMEN

En la actualidad, hay una tendencia al desarrollo de programas para uso empresarial en plataformas Web, dejando de lado el tradicional desarrollo de una aplicación de escritorio que debe ser copiada o distribuida en cada computador cliente. Esto trae consigo una enorme ventaja de mantener la aplicación siempre actualizada y corregida.

Una de estas plataformas Web más conocida es la del servicio Apache haciendo uso del lenguaje de programación PHP, instalados típicamente en un servidor con sistema operativo Linux. Este escenario de instalación comprende un conjunto de servicios que deben ser revisados bajo lineamientos de seguridad informática, puesto que el despliegue por defecto de dichos servicios tiene de por sí vulnerabilidades que pueden ser aprovechadas por los atacantes.

En este documento, se analiza un procedimiento típico de aseguramiento y fortalecimiento para un sistema de esta naturaleza perteneciente a una empresa de la ciudad de Guayaquil, el cual está siendo utilizado diariamente

para almacenamiento de archivos de trabajos terminados y por lo tanto amerita el nivel de seguridades que podamos lograr configurarle

ÍNDICE GENERAL

ABREVIATURAS Y SIMBOLOGÍA

FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
httpd	Servicio Apache
HTTPS	HTTP con encriptación
PHP	Hypertext Preprocessor
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
vsftpd	Servicio FTP de CentOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

MATRIFLEXO S.A. es una industria gráfica radicada en la ciudad de Guayaquil, con 19 años de trayectoria, dedicada a la producción y comercialización de planchas fotopolímeras para todo tipo de impresión flexográfica. Esta empresa genera una cantidad significativa de trabajos diarios, cada uno de los cuales mantiene archivos digitales que, tras ser entregados al cliente, deben ser almacenados en un sistema para su archivo histórico.

Así como MATRIFLEXO S.A., múltiples empresas mantienen históricos de archivos producto de sus actividades con sus clientes, los cuales necesitan ser referenciados o utilizados en posteriores trabajos, inclusive varios años después de haber sido originalmente realizados.

El sistema utilizado en MATRIFLEXO S.A. es un sistema desarrollado casa adentro, y su propiedad intelectual es propiedad de la misma empresa. Sin embargo, por ser un sistema basado en Web, con sistema operativo Linux y bases de datos open source, es un perfecto candidato para un hardening.

Adicionalmente, por la alta criticidad de la información digital generada en la empresa, se vuelve crucial implementar y/o asegurar que existan medidas de seguridad informática orientadas a preservar la confidencialidad de los archivos almacenados en el sistema.

CAPÍTULO 1

GENERALIDADES

1.1. Descripción del problema

El sistema que debemos realizar un aseguramiento es una típica aplicación Web que reside en un servidor Linux dentro de la red local. Este equipo consta de un servicio Apache, el cual sirve las peticiones HTTP de los clientes. Cada una de las páginas de la aplicación Web han sido desarrolladas en el lenguaje PHP, el cual es invocado por el servicio Apache para construir las respuestas para los usuarios. Los datos del sistema se almacenan en una base de datos PostgreSQL

El diagrama de contexto del sistema es el siguiente:

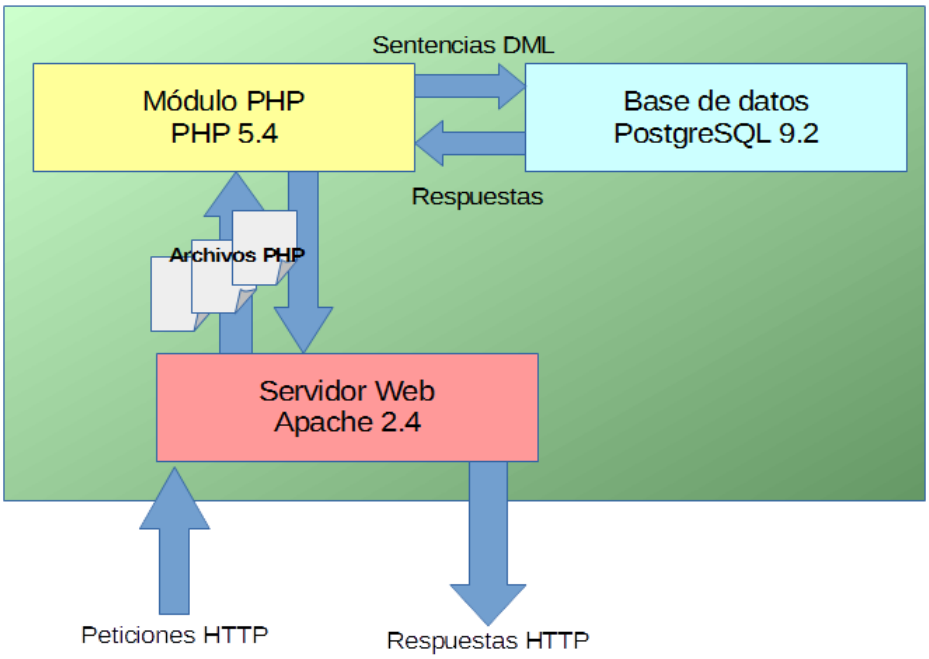

Figura 1.1. Diagrama de contexto del sistema

Las consideraciones del sistema son:

- Todos los roles del sistema, detallados en el diagrama de contexto, se ejecutan en un único servidor.
- El servidor tiene únicamente una dirección IP privada dentro del segmento de la red local de todas las computadoras de la empresa.

Las características físicas del sistema son las siguientes:

Tabla 1. Características físicas del sistema

Característica	Valor
Modalidad	Aplicación Web, clientes navegadores de

	Internet en la red local
Procesador	Intel Xeon E5-2650
Sistema operativo	Centos Linux 7.2.1511 64 bit
Versión de kernel	Linux 3.10.0-327
Aplicación Web	Apache 2.4.6
Lenguaje de desarrollo	PHP 5.4.16
Base de datos	PostgreSQL 9.2.15

1.2. Solución propuesta

Para suplir las necesidades de mejorar la seguridad informática del sistema organizacional utilizado, se partió de una estrategia similar a la del Cisco Security Wheel [1], la cual consta de 4 fases:

1. **Asegurar**: se identifican los objetivos de seguridad, se definen los factores de seguridad a los cuales vamos a realizar el levantamiento de situación actual y comenzamos su respectivo hardening.

2. **Monitorear**: se establecen mecanismos de monitoreo de logs y alertas del sistema, que permitan informar a las personas claves sobre incidentes en las seguridades implementadas

3. **Probar**: Se comprueba el correcto funcionamiento de las actividades de aseguramiento

4. **Mejorar**: Una vez establecida la línea base con las nuevas

políticas de seguridad, se definen recomendaciones y ajustes sobre las políticas implementadas

CAPÍTULO 2

METODOLOGÍA DE DESARROLLO DE LA SOLUCIÓN

2.1. Asegurar

En esta fase procedemos con la identificación de potenciales vulnerabilidades a la seguridad del software y comenzamos a tomar acción en cada una que encontremos.

2.1.1. Seguridad física

El primer nivel de aseguramiento que vamos a verificar en el sistema corresponde a la seguridad física del equipo servidor, para lo cual hemos encontrado los siguientes hallazgos y su

respectiva acción:

Hallazgo 2.1.1.a

Hallazgo: El BIOS no tiene una contraseña.

Acción tomada: Crear una contraseña para el BIOS.

Hallazgo 2.1.1.b

Hallazgo: El BIOS tiene como primera opción de boot por defecto, iniciar desde un DVD o CD

Acción tomada: Establecer el boot por defecto hacia el disco duro del sistema operativo

Hallazgo 2.1.1.c

Hallazgo: El BIOS permite el uso de dispositivos de almacenamiento externo (USB).

Acción tomada: Deshabilitar los dispositivos de almacenamiento externo en el BIOS.

2.1.2. Seguridad de sistema operativo

Los hallazgos encontrados en este ámbito, junto con su respectiva acción, se detallan a continuación

Hallazgo 2.1.2.a

Hallazgo: Múltiples módulos del sistema están desactualizados, desde la instalación del sistema operativo.

Acción tomada: Se ejecuta una actualización de todos los módulos instalados del sistema operativo, con el comando **yum update all** [1]

Hallazgo 2.1.2.b

Hallazgo: Existen algunos servicios levantados y configurados para inicio automático, que el servidor no necesita para el rol que está cumpliendo, como bind (servidor DNS) y smb

Acción tomada: Se deshabilitan los servicios públicos no necesarios, se dejan habilitados sólo y únicamente los servicios: ssh, httpd (apache), vsftpd y postgres aparte de los servicios esenciales de sistema (ej: firewalld, crond, etc.). Se configuran las reglas de firewalld para escuchar únicamente en los puertos de ssh y https [1]

Hallazgo 2.1.2.c

Hallazgo: Solo existe el usuario root. Con este usuario se inicia sesión en el servidor, cuando es necesario. Adicionalmente, no se ha cambiado la política de contraseñas desde que fue instalado el sistema operativo.

Acciones tomadas:

1. Se crean un usuario para tareas de administración “admin”. Se agrega al grupo de sudoers.

2. Se restringe a que root solamente inicie sesión desde el propio servidor físico, escribiendo solamente “tty1” en el archivo /etc/securetty
3. Se restringe a que el directorio /root solamente sea accesible para el usuario root, con el comando chmod 0700 /root
4. Verificamos que en CentOS 7 el algoritmo de hashing de password por defecto es SHA-512 con el comando “authconfig --test | grep hashing”
5. Configuramos el archivo /etc/security/pwquality.conf con las directivas: minlen=16, ucredit=-1, lcredit=-1, dcredit=-1, ocredit=-1 para fortalecer la complejidad de las contraseñas ingresadas por el usuario [1]
6. Configuramos el archivo /etc/login.defs con las directiva: PASS_MAX_DAYS 120, para requerir que el usuario deba cambiar su contraseña pasados 2 meses [1]

Hallazgo 2.1.2.d

Hallazgo: Se encuentra habilitado un servicio SSH para tareas de administración remotas, el cual está con pocas seguridades implementadas

Acciones tomadas:

1. Se exige utilizar Protocolo 2, con la directiva “Protocol 2”

dentro de /etc/ssh/sshd_config

2. Se restringe el acceso a root con las directivas “DenyUsers root”, “DenyGroups root” & “PermitRootLogin no” dentro de /etc/ssh/sshd_config [1]
3. Se configura para que clientes sin actividad por más de 5 minutos sean automáticamente desconectados, con las directivas “ClientAliveInterval 300” y “ClientAliveCountMax 5” dentro de /etc/ssh/sshd_config [1]
4. Se restringe el uso solamente al usuario admin, con la directiva “AllowUsers admin” dentro de /etc/ssh/sshd_config [1]
5. Se deshabilita passwords en blanco con la directiva “PermitEmptyPasswords no” dentro de /etc/ssh/sshd_config [1]

Hallazgo 2.1.2.e

Hallazgo: Se encuentra habilitado un servicio FTP, para subir cambios en los archivos fuente del sistema, a fin de implementar nuevas funcionalidades. El servicio FTP usa autenticación anónima

Acción tomada: Se implementa SFTP con el siguiente procedimiento:

1. Generamos un certificado auto-firmado con el comando:

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/vsftpd/sftp.key -out /etc/vsftpd/sftp.crt

2. Editamos el archivo /etc/vsftpd/vsftpd.conf para establecer las siguientes directivas:
    - ssl_enable=YES
    - allow_anon_ssl=NO
    - force_local_data_ssl=NO
    - force_local_logins_ssl=NO
    - ssl_tlsv1=YES
    - ssl_sslv2=NO
    - ssl_sslv3=NO
    - rsa_cert_file=/etc/vsftpd/sftp.crt
    - rsa_private_key_file=/etc/vsftpd/sftp.key
3. Reiniciamos el servicio vsftpd.

Con estas medidas implementadas, el servicio denegará las solicitudes de conexión anónimas y sin la encriptación TLS

2.1.3. Seguridad de base de datos

Los hallazgos encontrados en este ámbito, junto con su respectiva acción, se reflejan a continuación:

Hallazgo 2.1.3.a

Hallazgo: La aplicación Web se conecta a la base de datos sin

ninguna encriptación de por medio. Esto no ha sido prioridad debido a que tanto la aplicación Web como la base de datos están en el mismo servidor

Acción tomada: Implementar el protocolo SSL sería un paso innecesario, debido a que su función es eliminar los ataques de intercepción y un atacante que haya comprometido la aplicación Web ya comprometería la base de datos debido a que ambos roles residen en el mismo servidor. La solución preferible en este escenario es evitar que otros hosts se conecten a la base de datos, garantizar que el único cliente sea la aplicación Web. Esto lo hacemos con las siguientes verificaciones [4]:

1. Revisar que esté únicamente activada la conexión por medio de sockets locales en el archivo /var/lib/pgsql/data/pg_hba.conf; esto es, solo habilitar la línea “local <base-datos> <usuario> trust”
2. Revisar que el servicio no escuche en ninguna interface de red ethernet estableciendo la directiva listen_addresses = ' ‘ en el archivo /var/lib/pgsql/data/postgresql.conf

Hallazgo 2.1.3.b

Hallazgo: La aplicación Web se conecta a la base de datos utilizando el super-usuario por defecto “postgres”.

Acciones tomadas:

1. Crear un usuario específico para la aplicación Web con el comando “useradd <nombreusuario>”
2. Establecer una contraseña para el usuario con el comando “passwd <nombreusuario>”
3. Otorgar los privilegios necesarios para el usuario con los comandos (dentro de PostgreSQL): “grant connect on <base> to <nombreusuario>” , “grant all privileges on all tables in schema <esquema> to <nombreusuario>” , “grant usage on all views on <base> to <nombreusuario>” [4]
4. Crear un usuario específico para respaldos de la base de datos, siguiendo el procedimiento indicado anteriormente, pero otorgando solo privilegios de lectura (gran select on all tables) [4]

Hallazgo 2.1.3.c

Hallazgo: La base de datos de la aplicación tiene aún los privilegios por defecto para conexión para el usuario por defecto “public”

Acción tomada: Remover los privilegios por defecto con el comando “revoke connect on <base> from public” [4]

Hallazgo 2.1.3.d

Hallazgo: El logging está efectuándose según las reglas

puestas por el fabricante, esto es, directamente hacia stderr y con copia hacia el directorio pg_log, en 7 archivos, uno por cada día de la semana que se sobreescriben

Acción tomada: Establecer las siguientes directivas en el archivo /var/lib/pgsql/data/postgresql.conf para garantizar mayor cantidad de días de log [4]:

- log_destination = 'stderr,syslog'
- log_directory = '/var/log/postgresql/'
- loggin_collector = on (provoca la copia hacia el directorio /var/log/postgresql/)
- log_filename postgres-%d-%H%M%S.log
- log_truncate_on_rotation on
- log_rotation_age 1d
- log_rotation_size 50MB
- log_line_prefix '%t %c '

Esto creará archivos de log por cada día del mes, rotándolos cuando acabe el mes. En caso de que un log de un día supere los 50 MB, se creará otro archivo (para evitar archivos de log prácticamente inprocesables). Cada entrada de log estará precedida de la marca de tiempo (%t) y el ID de sesión (%c)

Hallazgo 2.1.3.e

Hallazgo: No se han establecido límites de tiempo para

ejecución de consultas y operaciones de modificación de datos, lo cual puede llevar a ataques de denegación de servicio y fugas de memoria

Acción tomada: Establecer las siguientes directivas en el archivo /var/lib/pgsql/data/postgresql.conf [4]:

- max_connections = 30
- statement_timeout = 60000
- log_min_duration_statement = 3000

Hallazgo 2.1.3.f

Hallazgo: No se han establecido auditorías de conexión y desconexión

Acción tomada: Establecer las siguientes directivas en el archivo /var/lib/pgsql/data/postgresql.conf [4]:

- log_connections = on
- log_disconnections = on

2.1.4. Seguridad de servidor web

Los hallazgos encontrados en este ámbito, junto con su respectiva acción, se reflejan a continuación

Hallazgo 2.1.4.a

Hallazgo: La directiva global “Listen” permite que Apache escuche en todas las interfaces de red que tiene o llegue a

tener el servidor

Acción tomada: Se elimina la directiva “Listen 80”. Debido a que el archivo de configuración /etc/httpd/conf.d/ssl.conf ya define una directiva “Listen 443”, estamos asegurados que el servidor escuchará en el puerto adecuado. (Más adelante configuraremos SSL) [3]

Hallazgo 2.1.4.b

Hallazgo: El directorio virtual para la aplicación web tiene la directiva “Allow from all”, la cual permite que cualquier host de la red acceda a la aplicación

Acción tomada: Establecer la directiva “Allow from <segmento de red>” [3] luego de definir cuál es el intervalo dentro del segmento de red en el cual están los hosts que van a acceder a la aplicación, excluyendo así otros dispositivos como impresoras, cámaras IP, etc.

Hallazgo 2.1.4.c

Hallazgo: No está configurada la directiva “ServerSignature off”, para evitar que las páginas de error contengan información de la versión de Apache que está ejecutando

Acción tomada: Establecer la directiva “ServerSignature off” [3]

Hallazgo 2.1.4.d

Hallazgo: No está configurada la directiva “TraceEnable off”, para prohibir las peticiones TRACE que a su vez pueden llevar a ataques de Cross-Site Tracing [3]

Acciones tomadas:

1. Establecer la directiva “TraceEnable off”
2. Establecer la directiva “<LimitExcept GET POST> deny from all </LimitExcept>” dentro del VirtualHost de la aplicación para reforzar aún más la escucha de únicamente los verbos GET y POST

Hallazgo 2.1.4.e

Hallazgo: No está configurada la directiva “ServerTokens ProductOnly”, para evitar que en el encabezado de las respuestas HTTP vaya incluido el sistema operativo del servidor

Acción tomada: Establecer la directiva “ServerTokens ProductOnly” [3]

Hallazgo 2.1.4.f

Hallazgo: Cuando el usuario escribe una ruta de directorio y en dicho directorio no existe un archivo index, la aplicación Web muestra un listado de los archivos en dicho directorio

Acción tomada: Es necesario establecer la directiva “Options -Indexes” en el VirtualHost de la aplicación Web

Hallazgo 2.1.4.g

Hallazgo: La aplicación Web no tiene encriptación. Todas las peticiones y respuestas HTTP viajan en texto plano, inclusive contraseñas. Este tráfico obviamente puede ser interceptado

Acción tomada: Habilitamos encriptación con el estándar TLS con un certificado auto-firmado, mediante los siguientes pasos: [3]

1. Generamos el certificado con el comando **openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/webapp.key -out /etc/ssl/certs/webapp.crt**
2. Agregamos dentro de la configuración del VirtualHost de la aplicación web las directivas
    - SSLProtocol –ALL +TLSv1 +TLSv1.1 +TLSv1.2
    - SSLCipherSuite ALL:!aNULL:!LOW:!MD5 (para utilizar solamente algoritmos de encriptación fuertes)
    - SSLCertificateFile /etc/ssl/certs/webapp.crt
    - SSLCertificateKeyFile /etc/ssl/private/webapp.key
    - SSLEngine on
3. Reiniciamos el servicio httpd

Hallazgo 2.1.4.h

Hallazgo: Cuando suceden errores en el lenguaje de

programación de la aplicación, se muestran errores en la respuesta HTTP. Adicionalmente, la respuesta HTTP muestra la versión de PHP en el encabezado.

Acción tomada: Establecer las siguientes directivas dentro de un nuevo archivo en la ubicación /etc/php.d/security.ini [3]:

- expose_php=Off
- display_errors=Off

Hallazgo 2.1.4.i

Hallazgo: Mod_Security es un módulo de seguridades para Apache, el cual no está instalado en el servidor. Este módulo protegen contra ataques de inyección de SQL, inyección de PHP, reconocimiento, inclusión de archivos remotos, cross-site scripting y directory-traversal

Acción tomada: Habilitamos el módulo en el servidor por medio de los siguientes pasos [3]:

1. Ejecutamos el comando **yum install mod_security**
2. Dentro del archivo /etc/httpd/conf.d/mod_security.conf verificamos que está establecida la directiva “SecFilterEngine On”
3. Reiniciamos el servicio httpd

Hallazgo 2.1.4.j

Hallazgo: Los archivos de log se almacenan en la ubicación por defecto, en un solo archivo de log para todos los virtual hosts y directorios web que pueda estar ejecutando el servidor Apache

Acción tomada: Redirigimos los log de error y acceso mediante las siguientes directivas dentro del VirtualHost de la aplicación [3]:

- ErrorLog “/var/log/webapp/error.log”
- CustomLog “/var/log/webapp/access.log” common

2.1.5. Seguridad de autenticación y autorización

Los hallazgos encontrados en este ámbito, junto con su respectiva acción, se detallan a continuación

Hallazgo 2.1.5.a

Hallazgo: La aplicación Web maneja dentro de su base de datos una tabla para almacenar los usuarios habilitados, y la contraseña de dichos usuarios se guarda como un hash MD5. Esto constituye una vulnerabilidad de ataque de fuerza bruta o por tablas de rainbow**.**

Acción tomada: La aplicación ha sido modificada para que los passwords se almacenen como un hash del nombre de usuario combinado con el password, y por último con un valor de salting establecido dentro de la aplicación, todo esto encriptado con el

algoritmo SHA256

Hallazgo 2.1.5.b

Hallazgo: La aplicación Web no registra en ningún log los intentos de inicio de sesión

Acción tomada: Configuramos un archivo de log para registrar los intentos de inicio de sesión, sean exitosos o no; mediante el siguiente procedimiento:

1. En /etc/rsyslog.conf definimos un archivo exclusivo para guardar los mensajes logeados en la facility local4, agregando la siguiente línea:

   local4.* /var/log/webapp/autorizaciones.log

2. En PHP registramos los mensajes que deseamos hacia la facility local4 con las siguientes sentencias:
   - **openlog (“webapp”, LOG_CONS, LOG_LOCAL4);** para abrir el log que escribiremos
   - **rsyslog(LOG_ERR , <mensaje>);** para inicios de sesión fallidos
   - **rsyslog(LOG_INFO , <mensaje>);** para inicios de sesión correctos

Hallazgo 2.1.5.c

Hallazgo: La aplicación Web maneja un sistema de

autorización por perfiles versus páginas, en el cual cada perfil está asignado si tiene o no autorización para peticiones a cada página del sistema.

Acción tomada: Si bien este mecanismo de autorización funciona adecuadamente para las necesidades de la aplicación, lo reforzamos con logs de qué usuario está intentando acceder a qué página y si fue o no exitoso, guardando en el mismo archivo y con las mismas condiciones del punto anterior

Hallazgo 2.1.5.d

Hallazgo: La aplicación Web permite pasar por alto la página de autenticación. Se puede escribir directamente una URL de una página del sistema, y el servidor la servirá

Acción tomada: En cada página PHP del servidor verificar que en las primeras líneas se detecte la existencia de una sesión:

```php
<?php
    session_start();
    if (!isset($_SESSION)) {
      header( "location: noautorizado.html" );
      die();
    }
```

Figura 2.1. Código PHP para redirigir por ausencia de sesión

2.1.6. Seguridad de respaldos

Los hallazgos encontrados en este ámbito, junto con su respectiva acción, se detallan a continuación

Hallazgo 2.1.6.a

Hallazgo: Los respaldos se efectúan manualmente con la intervención de un operador

Acción tomada: Se programa una tarea con el comando **at**, para que se ejecute todos los días a la medianoche un respaldo completo de la base de datos de la aplicación, manteniendo 30 días de histórico.

Hallazgo 2.1.6.b

Hallazgo: Los respaldos se almacenan en el mismo almacenamiento local del servidor

Acción tomada: Se programa dentro de la tarea anteriormente especificada, una tarea de copia del archivo de respaldos hacia un computador distinto dentro de la misma red, mediante el comando **scp.**

Hallazgo 2.1.6.a

Hallazgo: No hay un mecanismo de log para determinar si el respaldo se realizó correctamente o no

Acción tomada: Se programa dentro de la tarea anteriormente especificada, una tarea para registrar el resultado en los log del sistema operativo con los comandos:

- **logger -f /var/log/respaldos.log -p local0.info <mensaje>**

(Para respaldos exitosos)

- **logger -f /var/log/respaldos.log -p local0.err <mensaje>**

(Para respaldos exitosos)

2.2. Monitorear

En esta fase nos preocupamos de la revisión de los mecanismos de log que hemos implementado a lo largo de la fase anterior, verificando que son accesibles y pueden sernos útil para recabar información sobre la efectividad de las medidas implementadas

2.2.1. Monitoreo de logs

Los log de sistema para cada uno de los niveles de aseguramiento definidos son los siguientes

Tabla 2. Archivos de log por elemento asegurado

Elemento	**Archivo de log**
Peticiones de páginas	/var/log/webapp/access.log
Errores PHP suscitados	/var/log/webapp/error.log
Errores de la base de datos	/var/log/postgresql/*.log
Cumplimiento de los respaldos	/var/log/respaldos.log
Intentos de inicio de sesión Intentos de acceso a páginas	/var/log/webapp/autorizaciones.log

Los log pueden ser consultados por los operadores mediante

las herramientas de lectura como tail y grep

2.2.2. Monitoreo por medio de alertas

Para hacer más accesible la información para usuarios administrativos de la aplicación sobre intentos no autorizados de ingreso al sistema, se ha implementado un correo electrónico sobre alertas en dicho log, mediante el envío de los mensajes del log en la facility local4.

La implementación de esta funcionalidad conlleva una tarea adicional de configuración del servicio postfix, debido a que el módulo ommail presente en el servicio syslog no permite establecer credenciales para conectarse al servidor de correos. Por tanto, configuramos el servicio postfix (que sí permite establecer credenciales) como un relay hacia el servidor de correos, y a su vez hacemos que el syslog despache la solicitud de envío al posfix local.

El procedimiento completo sería de la siguiente manera:

1. Se crea un archivo /etc/rsyslog.d/email.conf con el siguiente contenido

Figura 2.2. Contenido de archivo /etc/rsyslog.d/email.conf

2. Instalamos Postfix con el comando **yum install postfix**

3. Establecemos las siguientes directivas en el archivo /etc/postfix/main.cf

   - mydomain = <dominio-correos>
   - mynetworks_style = host (Solamente recibir peticiones de envío de correo provenientes de este host)
   - relayhost = [<servidor-externo-correos>]:587
   - smtp_sasl_auth_enable = yes
   - smtp_sasl_security_options = noanonymous
   - smtp_sasl_password_maps=hash:/etc/postfix/sasl_passwd
   - smtp_use_tls = yes
   - smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt

4. Creamos un archivo /etc/postfix/sasl_passwd en el cual dejaremos establecidas las credenciales que usaremos

para conectarnos al servidor SMTP externo. El contenido de este archivo será una sola línea: **[<servidor-externo-correos>]:587 <usuario>:<contraseña>**

5. Convertimos el archivo generado en el paso anterior en un archivo de búsqueda para Postfix, con el comando **postmap /etc/postfix/sasl_passwd**. Esto generará en el mismo directorio un archivo sasl_passwd.db

6. Protegemos ambos archivos estableciéndoles su máscara de permisos con el comando **chmod 0600**

7. Iniciamos el servicio Postfix con el comando **service postfix start**

Con esta implementación, el servicio syslog derivará hacia el servicio local postfix el envío de un correo, el cual a su vez hará un relay hacia el servidor externo de correos. Como podemos darnos cuenta, se generará estos correos para el log con facility “local4”, es decir el log de autorizaciones y autenticaciones.

2.3. Probar

En esta fase ejecutamos pruebas sobre las medidas de aseguramiento tomadas anteriormente, y verificamos su efectividad

2.3.1. Pruebas de reconocimiento

Para realizar un reconocimiento de puertos abiertos, utilizamos el programa ZenMap, el cual es la versión con interface gráfica del programa **nmap** [5] para identificar los puertos abiertos en el servidor. Ejecutamos el comando para realizar un scan tipo connect, nivel 4 (agresivo) y con detección de servicios en puerto: **nmap -T4 -A <dirección IP del servidor>.** Como resultado, obtenemos que solo 2 puertos están abiertos:

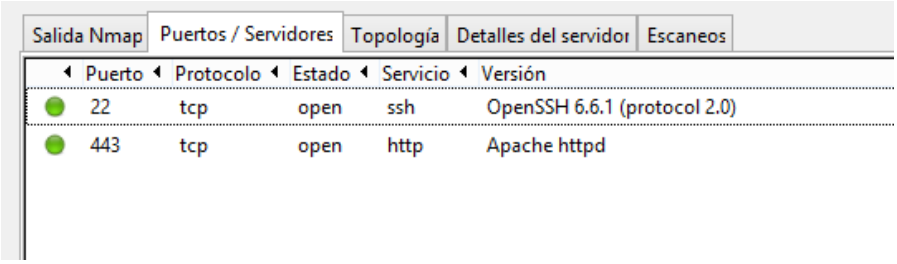

Figura 2.3. Resultado de NMap - puertos abiertos

Estos son los puertos 22, el cual dejamos habilitado el servicio ssh y el servicio sftp, y 443, el cual dejamos habilitado el servicio https.

La salida completa de la ejecución del comando la vemos en la siguiente figura:

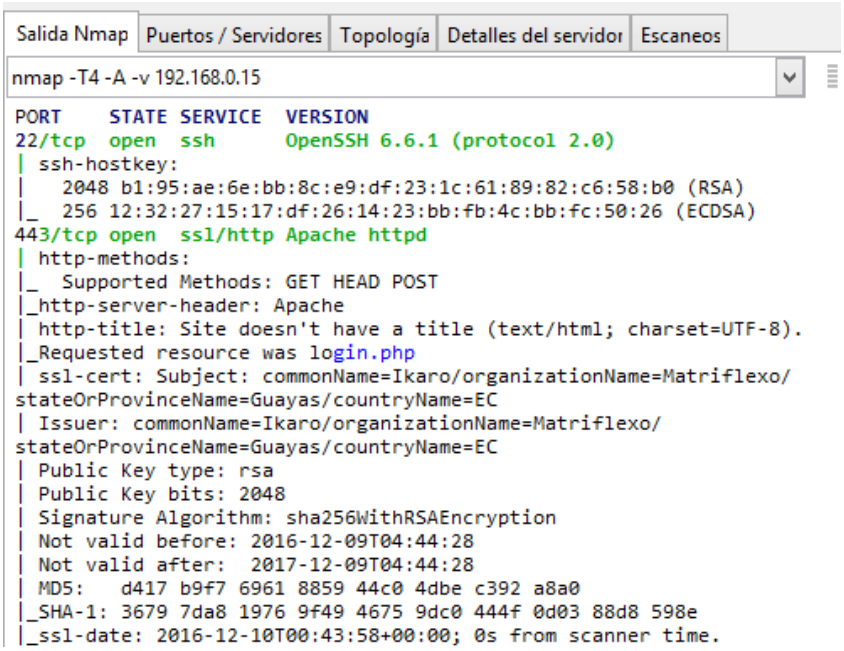

Figura 2.4. Resultado de NMap - Detección de servicios

En esta salida notamos que se detectó la ejecución de servicios ssh y http, así como que están protegidos por SSL. En Apache, vemos que solo permite los métodos GET, HEAD y POST, y que no arroja la versión del servicio

Con el pre-conocimiento de que el servidor ejecuta un servicio FTP, utilizamos el cliente de FTP FileZilla para intentar conectarnos al servicio FTP anónimamente y sin encriptación. Comprobamos que el servidor rechaza nuestro intento de conexión.

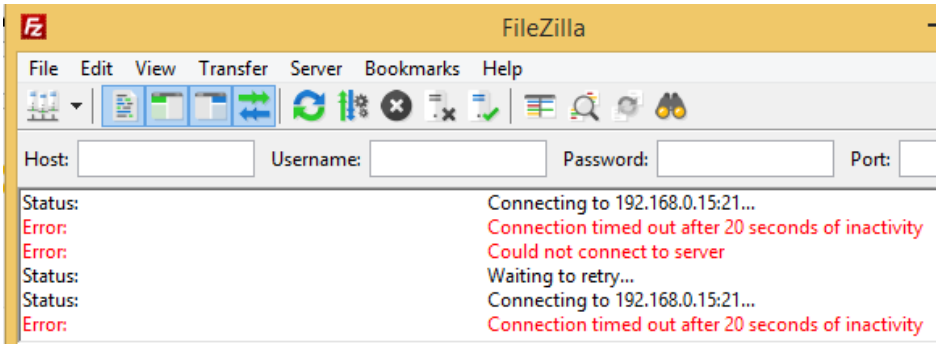

Figura 2.5. Conexión de FileZilla con usuario anónimo en FTP plano

Utilizamos el mismo cliente para intentar conectarnos ahora con SFTP, pero usuario anónimo. Comprobamos que el servidor rechaza nuestro intento de conexión

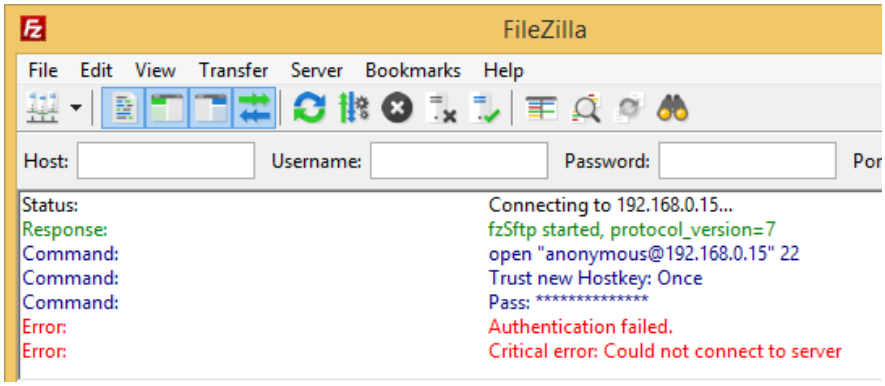

Figura 2.6. Conexión de FileZilla con usuario anónimo en FTP seguro

Por último, utilizamos el mismo cliente para conectarnos ahora con SFTP y con un usuario y contraseña. Comprobamos que el servidor aprueba nuestro intento de conexión

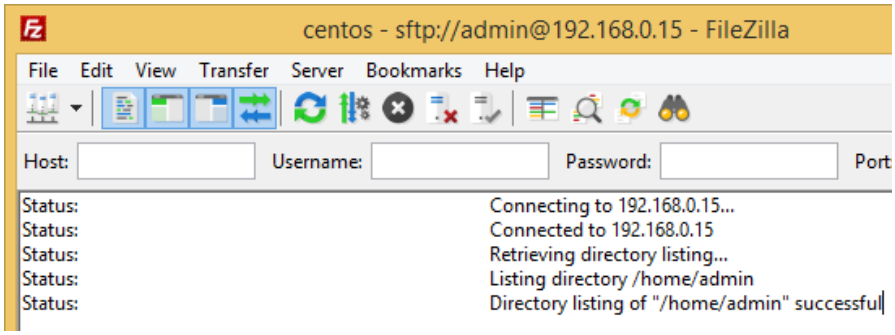

Figura 2.7. Conexión de FileZilla con usuario en FTP seguro

2.3.2. Pruebas de intercepción

Para realizar las pruebas de intercepción, escuchamos en la red con el programa Wireshark, el cual es capaz de analizar los datagramas intercambiados entre un cliente cualquiera y el servidor. Idealmente, dicho tráfico debería estar encriptado luego de haber aplicado las medidas de seguridad. El resultado de la intercepción se refleja en la siguiente figura

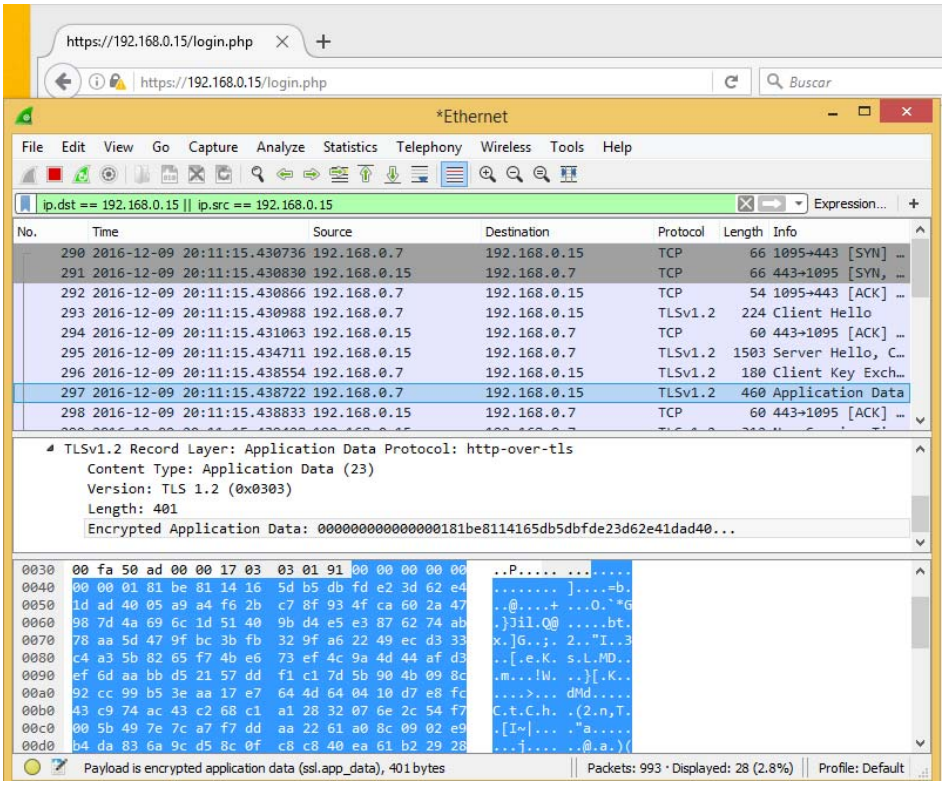

Figura 2.8. Resultados de Wireshark

Como podemos ver, la aplicación Web está utilizando el protocolo TLSv1.2, tal como habíamos especificado que no se permitan protocolos débiles. El contenido de las tramas está completamente encriptado, por lo tanto, hemos eliminado el riesgo de un ataque de intercepción sobre el intercambio de datos entre servidor y sus clientes.

2.3.3. Pruebas de generación

Para realizar las pruebas de generación, elaboramos una petición HTTP con datos inválidos hacia la aplicación Web. El primer efecto que observaremos es que, debido a que nuestra

aplicación Web tiene un rango de direcciones IP clientes a las cuales está permitido atender, debería denegar nuestra petición. En segundo lugar, nuestra aplicación Web debería rechazar las peticiones si no está definida una sesión de usuario.

Solicitamos el acceso a la aplicación desde un navegador de Internet en un host no autorizado y observamos la respuesta

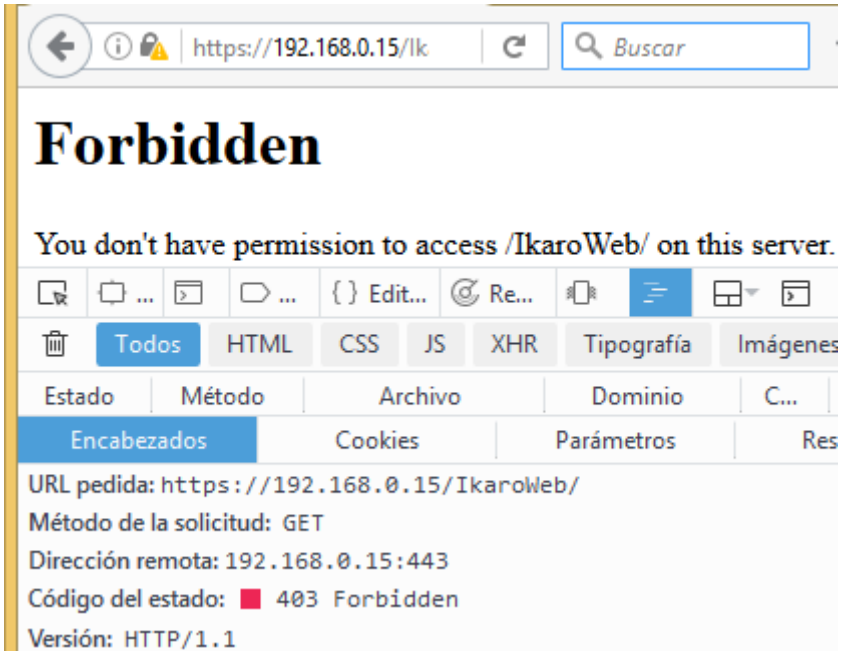

Figura 2.9. Rechazo de Apache por dirección IP de cliente

Como podemos observar, el servidor ha denegado la petición del recurso Web. Ahora desde un host autorizado intentaremos solicitar un recurso sin haber iniciado sesión previamente en el sistema

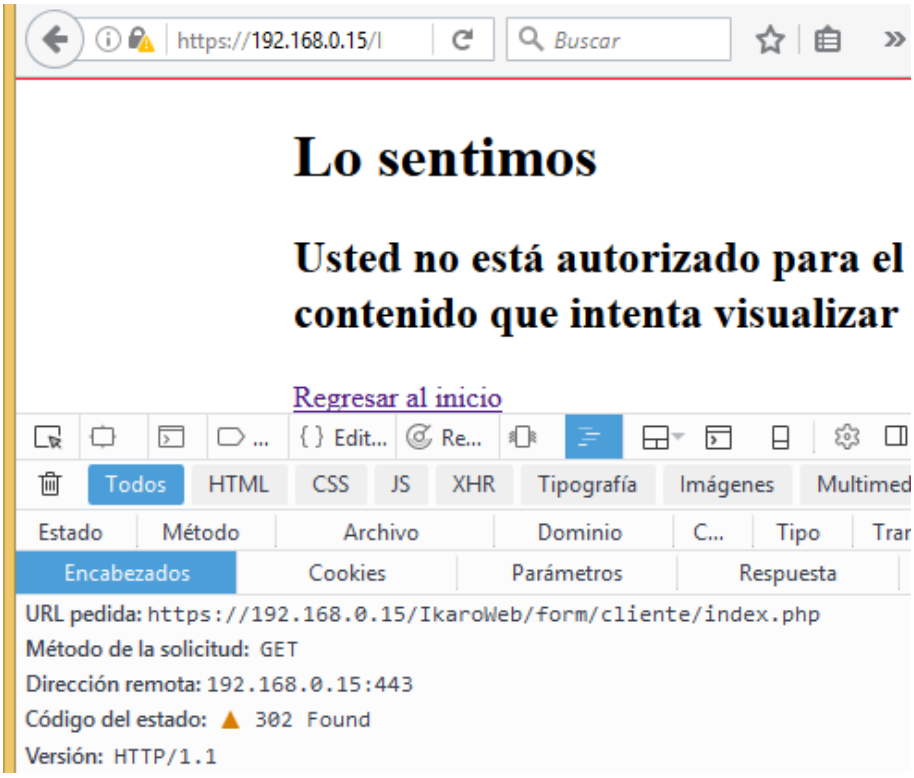

Figura 2.10. Rechazo de Apache por falta de sesión

Como podemos evidenciar, hemos intentado acceder sin antes haber pasado por el proceso de login. El sistema emite una respuesta con código HTTP 302, el cual equivale al estado “Found” y siempre incluye una cabecera Location, en la cual especifica una URL al cual el cliente debe redirigirse. Esta URL resulta ser una página estática que muestra el mensaje de usuario no autorizado.

CAPÍTULO 3

ANÁLISIS DE RESULTADOS

3.1. Resumen de seguridades implementadas

Un resumen de las seguridades implementadas en la fase de Asegurar se detalla en la siguiente tabla:

Tabla 3. Seguridades implementadas por componente del sistema

Componente del sistema	**Seguridades implementadas**
Seguridad física	Protección de manipulación del BIOS y secuencia de boot
Sistema operativo	Eliminación de servicios y cierre de puertos

	innecesarios Protección del superusuario root Hardening de políticas de contraseñas Hardening de servicio ssh Hardening de servicio ftp
Base de datos	Eliminación de protocolo TCP Protección del superusuario postgres Eliminación de permisos por defecto Afinamiento de logs
Servidor Web	Implementación de encriptación de tráfico Ocultación de detalles del servidor y del lenguaje de programación PHP Implementación de módulo de seguridades Afinamiento de logs
Autenticación y autorización	Hardening de encriptación Afinamiento de logs
Respaldos	Automatización de respaldos Afinamiento de logs

3.2. Evaluación con respecto a la situación inicial

Con las seguridades implementadas podemos evaluar los riesgos que han sido mitigados en los siguientes aspectos:

1. **Riesgo de interrupción**: Las restricciones puestas en seguridad física, junto con las restricciones de superusuarios en el sistema operativo y en la base de datos, aminoran la probabilidad de ocurrencia de que un atacante tome el control del servicio y decida suspenderlo.

2. **Riesgo de intercepción y modificación**: La encriptación del tráfico de la aplicación Web aminora el riesgo de que un atacante logre o modificar obtener la información que se intercambia entre

el servidor y sus clientes. Así mismo el cierre de puertos innecesarios del servidor impide que atacantes aprovechen estas puertas de ingreso.

3. **Riesgo de generación**: Las restricciones de autenticación y autorización permiten proteger el sistema de peticiones falsas o inválidas, que a su vez puedan provocar daños mayores a la integridad de la aplicación

3.3. Tendencias para mejorar

La última fase del Cisco Security Wheel [1] consiste en proponer mejoras periódicas a las implementaciones de seguridad. En este caso hay varios aspectos que pueden mejorarse sobre el proyecto aquí presentado:

1. **Separar roles en distintos servidores**. Debido a una restricción del cliente, todos los roles del sistema residen en un mismo servidor físico, el cual rápidamente es identificado como un único punto de fallo, factor crítico en la continuidad del negocio.

2. **Mantener un sistema espejo**. Ante un evento catastrófico, sería ideal mantener la continuidad de operaciones del negocio con un sistema espejo que permanece inactivo hasta que se detecte que el sistema principal ha sido comprometido.

3. **Revisar periódicamente logs**. Aunque es una tarea repetitiva que llega a convertirse en monótona, la revisión periódica de logs puede ayudar a descubrir nuevas vulnerabilidades que atacantes podrían estar aprovechando.

4. **Mantener el servidor actualizado**. Los diversos módulos que componen la distribución de Linux que hemos utilizado constantemente liberan actualizaciones de seguridad. Es importante mantener el servidor actualizado con los últimos parches de seguridad.

CONCLUSIONES Y RECOMENDACIONES

1. Hemos encontrado problemas básicos como transmisión de datos sensibles en texto plano, puertos abiertos innecesariamente y falta de actualización, las cuales hemos podido mitigar con las técnicas de aseguramiento aprendidas. Esto se debe probablemente a metas de tiempo muy estrictas impuestas para la implementación y puesta en marcha de cualquier aplicación de uso empresarial, o a una despreocupación de seguridad en las fases iniciales del proyecto

2. Se han implementado medidas de eliminación de las vulnerabilidades más populares sobre el típico sistema compuesto por un servidor Web accediendo a una base de datos local. No hemos llegado

probablemente al nivel óptimo de seguridad, pero sí a un nivel aceptable de eliminación y mitigación de riesgos más comunes.

3. Seguir la metodología del Cisco Security Wheel nos ha ayudado a realizar nuestro fortalecimiento de las seguridades del servidor en una forma ordenada y efectiva, dejando la puerta abierta a un ciclo constante de mejora

4. Cumplir con el ciclo constante del Cisco Security Wheel, a través de la fase de Mejorar, en la cual se descubren nuevas potenciales vulnerabilidades y oportunidades para reforzar la seguridad.

5. Dependiendo de la disponibilidad del cliente, también sería útil la separación de roles en servidores físicos o virtuales, a fin de distribuir el punto de falla único existente hasta ahora.

6. Es imperativo mantener actualizado los servicios instalados y el sistema operativo, para ir cubriendo las vulnerabilidades que se van descubriendo día a día.

BIBLIOGRAFÍA

[1] Cisco Systems Inc., Fundamentals of Network Security v1.1, http://www.cisco.com/web/learning/netacad/demos/FNSDemo1_1/ch1/1_3_1/, fecha de consulta 1 de diciembre de 2016, capítulo 1.3.1.

[2] Red Hat, Inc., Red Hat Enterprise Linux 7 Security Guide. https://access. redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/pdf/ Security_ Guide/Red_Hat_Enterprise_Linux-7-Security_Guide-en-US.pdf, fecha de consulta 1 de diciembre de 2016

[3] The Apache Software Foundation, Apache HTTP Server Version 2.4 Documentation, http://www-us.apache.org/dist/httpd/docs/httpd-docs-2.4.23.en.pdf, fecha de consulta 1 de diciembre de 2016

[4] The PostgreSQL Global Development Group (2016). PostgreSQL 9.2.19 Documentation, https://www.postgresql.org/files/documentation/pdf/9.2/ postgresql-9.2-A4.pdf, fecha de consulta 1 de diciembre de 2016

[5] Gordon “Fyodor” Lyon (2011) NMap Reference Guide, https://linux.die.net/man/1/nmap, fecha de consulta 1 de diciembre de 2016